


USN

--	--	--	--	--	--	--	--	--	--

Internal Assessment Test I – August 2023

Sub:	Web Technologies						Sub Code:	22MCA24	
Date:	2-08-2023	Duration:	90 min's	Max Marks:	50	Sem:	II	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

No.	Question	MARKS	OBE	
			CO	RBT
PART I				
1	a) What is web server? Explain its file structure. b) Briefly explain the following i) Web browsers ii) URL	[10]	CO2,CO7	L2
OR				
2	Explain request phase and response phase of HTTP	[10]	CO2,CO7	L3
PART II				
3	Explain HTML 5 progress bar and Media Elements (<audio> & <video>)	[10]	CO2,CO7	L2
OR				
4	Explain different types of lists in XHTML with example. Explain nested list with example	[10]	CO2,CO7	L3
PART III				
5	Create a student registration form to accept name, gender, date of birth, qualification, address and pincode. Provide reset and submit buttons.	[10]	CO2,CO7	L4
OR				
6	Create frames as shown in the figure using frameset tag 	[10]	CO2,CO7	L4
PART IV				
7	Describe various selector forms with suitable examples.	[10]	CO2,CO7	L3
OR				
8	Discuss all the CSS Font properties.	[10]	CO2,CO7	L3
PARTV				
9	Explain box model of CSS.	[10]	CO2,CO7	L3
OR				
10	Explain different ways of including CSS style information to a HTML doc.	[10]	CO2,CO7	L3

-
- 1a)
a) What is web server? Explain its file structure.
WEB SERVERS
Web servers are programs that provide documents to requesting browsers. Example: Apache
- Web server operations:
 All the communications between a web client and a web server use the HTTP
 When a web server begins execution, it informs the OS under which it is running & it runs as a background process
 A web client or browser, opens a network connection to a web server, sends information requests and possibly data to the server, receives information from the server and closes the connection.

☒ The primary task of web server is to monitor a communication port on host machine, accept HTTP commands through that port and perform the operations specified by the commands.

☒ When the URL is received, it is translated into either a filename or a program name

General characteristics of web server:

☒ The file structure of a web server has two separate directories

☒ The root of one of these is called document root which stores web documents

☒ The root of the other directory is called the server root which stores server and its support software's

☒ The files stored directly in the document root are those available to clients through top level URLs

☒ The secondary areas from which documents can be served are called virtual document trees.

☒ Many servers can support more than one site on a computer, potentially reducing the cost of each site and making their maintenance more convenient. Such secondary hosts are called virtual hosts.

☒ Some servers can serve documents that are in the document root of other machines on the web; in this case they are called as proxy servers

1b) Briefly explain the following i) Web browsers ii) URL

WEB BROWSERS

- Documents provided by servers on the Web are requested by browsers, which are programs running on client machines.
- They are called browsers because they allow the user to browse the resources available on servers.
- Mosaic was the first browser with a graphical user interface.
- A browser is a client on the Web because it initiates the communication with a server, which waits for a request from the client before doing anything.
- In the simplest case, a browser requests a static document from a server.
- The server locates the document among its servable documents and sends it to the browser, which displays it for the user.
- Sometimes a browser directly requests the execution of a program stored on the server.
- The output of the program is then returned to the browser.
- Examples: Internet Explorer, Mozilla Firefox, Netscape Navigator, Google Chrome, Opera etc.,

UNIFORM RESOURCE LOCATORS

- Uniform Resource Locators (URLs) are used to identify different kinds of resources on Internet.
 - If the web browser wants some document from web server, just giving domain name is not sufficient because domain name can only be used for locating the server.
 - It does not have information about which document client needs. Therefore, URL should be provided.
 - The general format of URL is: scheme: object-address
 - Example: http: www.vtu.ac.in/results.php
 - The scheme indicates protocols being used. (http, ftp, telnet...)
 - In case of http, the full form of the object address of a URL is as follows: //fully-qualified-domain-name/path-to-document
 - URLs can never have embedded spaces
 - It cannot use special characters like semicolons, ampersands and colons
 - The path to the document for http protocol is a sequence of directory names and a filename, all separated by whatever special character the OS uses. (Forward or backward slashes)
 - The path in a URL can differ from a path to a file because a URL need not include all directories on the path
 - A path that includes all directories along the way is called a complete path
 - Example: <http://www.gumboco.com/files/f99/storefront.html>
 - In most cases, the path to the document is relative to some base path that is specified in the configuration files of the server. Such paths are called partial paths.
 - Example: <http://www.gumboco.com/storefront.html>
-
-

2. Explain request phase and response phase of HTTP

THE HYPERTEXT TRANSFER PROTOCOL

- HTTP contains two phases request phase and response phase.
- Each HTTP communication between browser and server consist of two part, a header and a body, header contain information about communication and body contain data or message of the communication if there is any.

Request Phase: The general form of an HTTP request is as follows:

1. HTTP method Domain part of the URL HTTP version
2. Header fields
3. Blank line
4. Message body

The following is an example of the first line of an HTTP request: GET /storefront.html HTTP/1.1

Table 1.1 HTTP request methods

Method	Description
GET	Returns the contents of the specified document
HEAD	Returns the header information for the specified document
POST	Executes the specified document, using the enclosed data
PUT	Replaces the specified document with the enclosed data
DELETE	Deletes the specified document

The format of a header field is the field name followed by a colon and the value of the field. There are four categories of header fields:

1. General: For general information, such as the date
2. Request: Included in request headers
3. Response: For response headers
4. Entity: Used in both request and response headers A wildcard character, the asterisk (*), can be used to specify that part of a MIME type can be anything

The Response Phase:

The general form of an HTTP response is as follows:

1. Status line
2. Response header fields
3. Blank line
4. Response body The status line includes the HTTP version used, a three-digit status code for the response, and a short textual explanation of the status code. For example, most responses begin with the following: HTTP/1.1 200 OK The status codes begin with 1, 2, 3, 4, or 5. The general meanings of the five categories specified by these first digits are shown in Table 1.2

Table 1.2 First digits of HTTP status codes

First Digit	Category
1	Informational
2	Success
3	Redirection
4	Client error
5	Server error

3) Explain HTML 5 progress bar and Media Elements (<audio> & <video>)

HTML <progress> tag is used to display the progress of a task. It provides an easy way for web developers to create progress bar on the website. It is mostly used to show the progress of a file uploading on the web page. The HTML progress tag is new in HTML5 so you must use new browsers.

```
<label for="file">Downloading progress:</label>
<progress id="file" value="32" max="100"> 32% </progress>
```

Attribute	Value	Description
max	number	Specifies how much work the task requires in total. Default value is 1
value	number	Specifies how much of the task has been completed

The HTML <audio> Element

To play an audio file in HTML, use the <audio> element:

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

The HTML <video> element is used to show a video on a web page.

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

4) Explain different types of lists in XHTML with example. Explain nested list with example

Lists

1) Unordered List

The tag, which is a block tag, creates an unordered list. Each item in a list is specified with an tag (li is an acronym for list item). Any tags can appear in a list item, including nested lists. When displayed, each list item is implicitly preceded by a bullet.

```

<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- unordered.html
  An example to illustrate an unordered list
  -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Unordered list </title>
  </head>
  <body>
    <h3> Some Common Single-Engine Aircraft </h3>
    <ul>
      <li> Cessna Skyhawk </li>
      <li> Beechcraft Bonanza </li>
      <li> Piper Cherokee </li>
    </ul>
  </body>
</html>

```

Figure 2.15 shows a browser display of `unordered.html`.

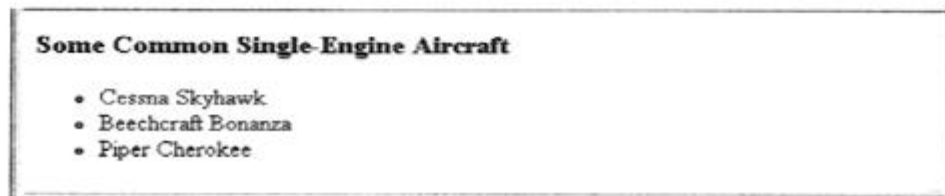


Figure 2.15 Display of `unordered.html`

2) Ordered List

Ordered lists are lists in which the order of items is important. This ordered-ness of a list is shown in the display of the list by the implicit attachment of a sequential value to the beginning of each item. The default sequential values are Arabic numerals, beginning with 1. An ordered list is created within the block tag `ol`. The items are specified and displayed just as are those in unordered lists, except that the items in an ordered list are preceded by sequential values instead of bullets.

```

<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- ordered.html
  An example to illustrate an ordered list
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Ordered list </title>
  </head>
  <body>
    <h3> Cessna 210 Engine Starting Instructions </h3>
    <ol>
      <li> Set mixture to rich </li>
      <li> Set propeller to high RPM </li>
      <li> Set ignition switch to "BOTH" </li>
      <li> Set auxiliary fuel pump switch to "LOW PRIME" </li>
      <li> When fuel pressure reaches 2 to 2.5 PSI, push
        starter button
      </li>
    </ol>
  </body>
</html>

```

Figure 2.16 shows a browser display of ordered.html.

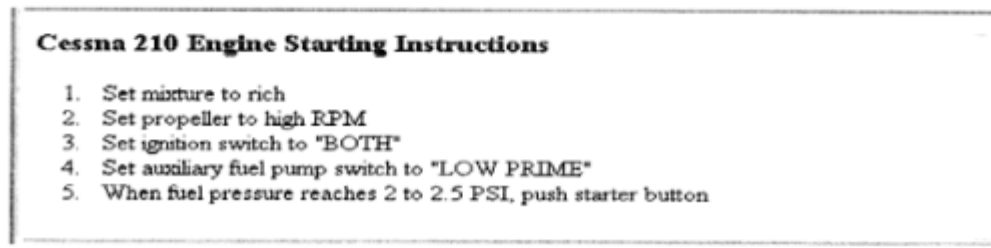


Figure 2.16 Display of ordered.html

Nested Lists:

1. Information Science
 1. OOMD
 2. Java & J2ee
 - classes and methods
- exceptions
- applets
- servlets
- Computer Networks
 - Part 1
 - Part 2
- DBMS
- Operations Research
- Computer Science
 1. Compiler Design
 2. FLAT
 - NFA
 - DFA
 - CFG

3. Computer Graphics
4. Artificial Intelligence

-
1. Information Science
 1. OOMD
 2. Java & J2ee
 - classes and methods
 - exceptions
 - applets
 - servelets
 3. Computer Networks
 - Part 1
 - Part 2
 4. DBMS
 5. Operations Research
 2. Computer Science
 1. Compiler Design
 2. FLAT
 - NFA
 - DFA
 - CFG
 3. Computer Graphics
 4. Artificial Intelligence

3) Definition List As the name implies, definition lists are used to specify lists of terms and their definitions, as in glossaries. A definition list is given as the content of a tag, which is a block tag. Each term to be defined in the definitionlist is given as the content of a tag. The definitions themselves are specified as the content of tags. The defined terms of a definition list are usually displayed in the left margin; the definitions are usually shown indented on the line or lines following the term.

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- definition.html
  An example to illustrate definition lists
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Definition lists </title>
  </head>
  <body>
    <h3> Single-Engine Cessna Airplanes </h3>
    <dl>
      <dt> 152 </dt>
      <dd> Two-place trainer </dd>
      <dt> 172 </dt>
      <dd> Smaller four-place airplane </dd>
      <dt> 182 </dt>
      <dd> Larger four-place airplane </dd>
      <dt> 210 </dt>
      <dd> Six-place airplane - high performance </dd>
    </dl>
  </body>
</html>
```

Single-Engine Cessna Airplanes

152	Two-place trainer
172	Smaller four-place airplane
182	Larger four-place airplane
210	Six-place airplane - high performance

5) Create a student registration form to accept name, gender, date of birth, qualification, address and pincode. Provide reset and submit buttons.

```
<!-- Html Document Begins-->
<!DOCTYPE html>
<html>
  <!-- Header Section-->
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>
  Student Registration Form
</title>
</head>

  <!--Body of the Webpage-->
<body bgcolor="orange">

  <!--Start of Form-->
  <div style="margin: auto; width: 30%;">
  <form>
    <h2>Student Registration Form</h2>
    <p>Fill in this form to register</p>
    <br>

  <!--Input elemets for form-->
    <label><b>First Name</b></label>
    <input type="text" placeholder="Enter your first name" name="first_name" required>
    <br>

    <br>
    <label><b>Last Name</b></label>
    <input type="text" placeholder="Enter your last name" name="last_name" required>
    <br>
    <br>
    <label><b>E-mail</b></label>
    <input type="email" placeholder="Enter your e-mail" name="email" required>
    <br>
    <br>
    <label><b>Date of Birth</b></label>
    <input type="date" name="dob" required>
    <br>
    <br>
    <label><b>Set Username</b></label>
```



```

<input type="text" placeholder="Set Username" name="username" required>
<br>
<br>
<label><b>Set Password</b></label>
<input type="password" placeholder="Set password" name="password" required>
<br>

<br>
<label><b>Gender</b></label><br>
<input type="radio" name="gender" value="Male">
<label for="Male">Male</label><br>
<input type="radio" name="gender" value="Female">
<label for="Female">Female</label><br>
<input type="radio" name="gender" value="Others">
<label for="Others">Others</label>
<br>

<br>
<label><b>Course :</b></label>
  <select>
    <option value="Course">Course</option>
    <option value="CS">Computer Fundamentals</option>
    <option value="AI">Artificial Intelligence</option>
    <option value="ML">Machine Learning</option>
    <option value="OOPS">Object Oriented Programming</option>
    <option value="DBMS">Database Management System</option>
  </select>
<br>
<br>
<input type="button" value="Register"/>
</form>
</div>
</body>
</html>

```

6) Create frames as shown in the figure using frameset tag



```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
</head>

```

```

<frameset cols="20%,*">
  <frame src="1.html"/>
  <frameset rows="10%,30%,*">
    <frame src="1.html"/>
    <frame src="2.html"/>
    <frameset cols="50%,*">
      <frame src="1.html"/>
      <frame src="2.html"/>
    </frameset>
  </frameset>
</frameset>
</html>

```

7) Describe various selector forms with suitable examples.

SELECTOR FORMS

1) Simple Selector Forms:

In case of simple selector, a tag is used. If the properties of the tag are changed, then it reflects at all the places when used in the program. The selector can be any tag. If the new properties for a tag are not mentioned within the rule list, then the browser uses default behaviour of a tag.

Eg: h1 { font-size : 24pt; }
 h2, h3 { font-size : 20pt; }
 body b em { font-size : 14pt; }

Only applies to the content of 'em' elements that are descendent of bold element in the body of the document. This is a contextual selector

2) Class Selectors: Class selectors are used to allow different occurrences of the same tag to use different style specifications.

Eg

```
<head>
```

```
  <style type = "text/css">
```

```
    p.one { font-family: 'Lucida Handwriting'; font-size: 25pt; color: Red; }
```

```
    p.two{ font-family: 'Monotype Corsiva'; font-size: 50pt; color: green; }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <p class = "one">Web Technology</p>
```

```
  <p class = "two">Web Technology</p>
```

```
</body>
```

3) Generic Selectors: Sometimes it is convenient to have a class of Style specification that applies to the content of more than one kind of tag. This is done by using a generic class, which is defined without a tag name in its name. In place of the tag name, you use the name of the generic class, which must begin with a period.

Eg

```
<head>
```

```
  <style type = "text/css">
```

```
    .sale{ font-family: 'Monotype Corsiva'; color: green; }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <p class = "sale">Weekend Sale</p>
```

```
  <h1 class = "sale">Weekend Sale</h1>
```

```
  <h6 class = "sale"> Weekend Sale</h6>
```

```
</body>
```

4) id Selectors: An id selector allows the application of a style to one specific element.

Eg:

<head>

<style type = "text/css">

#one { font-family: 'Lucida Handwriting'; font-size: 25pt; color: Red; }

#two { font-family: 'Monotype Corsiva'; font-size: 50pt; color: green; }

</style>

</head>

<body>

<p id = "one">Web Technology</p>

<p id = "two">Web Technology</p>

</body>

5) Universal Selectors: The universal selector, denoted by an asterisk (*), applies its style to all elements in a document.

<head>

<style type = "text/css">

***{ font-family: 'Lucida Handwriting'; font-size: 25pt; color: Red; }**

</style>

</head>

<body>

<p>Web Technology</p>

<p>Web Technology</p>

</body>

6) Pseudo Classes: Pseudo class selectors are used if the properties are to be changed dynamically. For example: when mouse movement happens, in other words, hover happens or focus happens.

```
<head>
```

```
  <style type = "text/css">
```

```
    input:focus { font-family: 'lucida calligraphy'; color: purple; font-size:100; }
```

```
    input:hover { font-family: 'lucida handwriting';color: violet; font-size:40; }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <form action = " ">
```

```
    <p><label> NAME: <input type = "text" /></label></p>
```

```
  </form>
```

```
</body>
```

8) Discuss all the CSS Font properties.

FONT PROPERTIES

1) Font Families:

The font-family property is used to specify a list of font names. The browser uses the first font in the list that it supports.

For example, the property: font-family: Arial, Helvetica, Futura tells the browser to use Arial if it supports that font.

If not, it will use Helvetica if it supports it.

If the browser supports neither Arial nor Helvetica, it will use Futura if it can. If the browser does not support any of the specified fonts, it will use an alternative of its choosing. If a font name has more than one word, the whole name should be delimited by single quotes, as in the following example: font-family: 'Times New Roman'.

2) Font Sizes: The font-size property does what its name implies. For example, the following property specification sets the font size for text to 10 points:

```
font-size: 10pt
```

Many relative font-size values are defined, including xx-small, x-small, small, medium, large, x-large, and xx-large. In addition, smaller or larger can be specified. Furthermore, the value can be a percentage relative to the current font size.

3) Font Variants: The default value of the font-variant property is normal, which specifies the usual character font. This property can be set to small-caps to specify small capital characters. These characters are all uppercase, but the letters that are normally uppercase are somewhat larger than those that are normally lowercase.

4) Font Styles: The font-style property is most commonly used to specify italic, as in font-style: italic

5) Font Weights: The font-weight property is used to specify the degree of boldness, as in font-weight: bold. Besides bold, the values normal, bolder, and lighter can be specified. Specific numbers also can be given in multiples of 100 from 100 to 900, where 400 is the same as normal and 700 is the same as bold.

6) Font Shorthands: If more than one font property must be specified, the values can be stated in a list as the value of the font property. The order in which the property values are given in a font value list is important. The order must be as follows: The font names must be last, the font size must be second to last, and the font

style, font variant, and font weight, when they are included, can be in any order but must precede the font size and font names. font: bold 14pt 'Times New Roman'

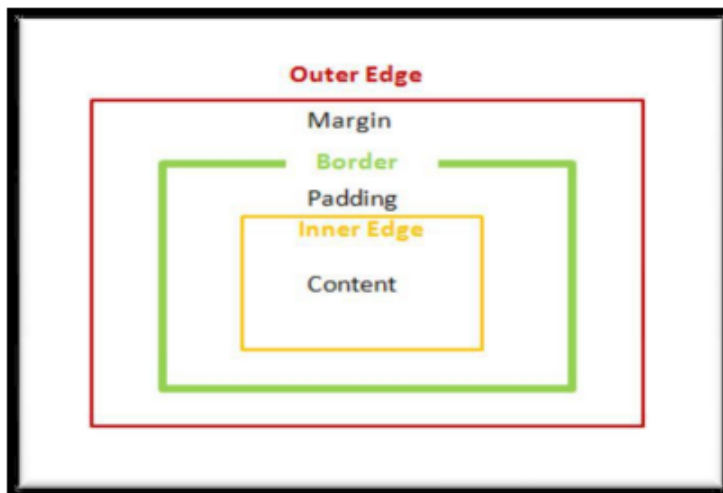
```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> Font properties </title>
    <style type = "text/css">
      p.major {font-size: 14pt;
                font-style: italic;
                font-family: 'Times New Roman';
              }
      p.minor {font: 10pt bold 'Courier New';}
      h2 {font-family: 'Times New Roman';
          font-size: 24pt; font-weight: bold}
      h3 {font-family: 'Courier New'; font-size: 18pt}
    </style>
  </head>
  <body>
    <p class = "major">
      If a job is worth doing, it's worth doing right.
    </p>
    <p class = "minor">
      Two wrongs don't make a right, but they certainly
      can get you in a lot of trouble.
    </p>
    <h2> Chapter 1 Introduction </h2>
    <h3> 1.1 The Basics of Computer Networks </h3>
  </body>
</html>
```

7) Text Decoration: The text-decoration property is used to specify some special features of text. The available values are line-through, overline, underline, and none, which is the default.

9) Explain box model of CSS.

THE BOX MODEL

- On a given web page or a document, all the elements can have borders.
- The borders have various styles, color and width.
- The amount of space between the content of the element and its border is known as *padding*.
- The space between border and adjacent element is known as *margin*.



Borders:

1) Border-style property controls whether the elements content has a border, as well as the style of the border

It can be dotted, dashed, double

The styles of one of the four sides of an element can be set with

- o Border-top-style
- o Border-bottom-style
- o Border-left-style
- o Border-right-style

2) Border-width is used to specify the thickness of a border

It can be thin, medium, thick or any length value

The width of each of the four borders of an element specified with:

- o Border-top-width
- o Border-bottom-width
- o Border-left-width
- o Border-right-width

3) Border-color control color of a border

The width of each of the four borders of an element specified with:

- o Border-top-color
- o Border-bottom-color
- o Border-left-color
- o Border-right-color

```

<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- borders.html
An example of a simple table with various borders
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head> <title> Table borders </title>
<style type = "text/css">
table {border-top-width: medium;
border-bottom-width: thick;
border-top-color: red;
border-bottom-color: blue;
border-top-style: dotted;
border-bottom-style: dashed;
}
p {border-style: dashed; border-width: thin;
border-color: green
}
</style>
</head>
<body>
<table border = "5">
<caption> Fruit Juice Drinks </caption>
<tr>
<th> </th>
<th> Apple </th>
<th> Orange </th>
<th> Screwdriver </th>
</tr>
<tr>
<th> Breakfast </th>
<td> 0 </td>
<td> 1 </td>
<td> 0 </td>
</tr>

<tr>
<th> Lunch </th>
<td> 1 </td>
<td> 0 </td>
<td> 0 </td>
</tr>
<tr>
<th> Dinner </th>
<td> 0 </td>
<td> 0 </td>
<td> 1 </td>
</tr>
</table>
<p>
Now is the time for all good Web programmers to
learn to use style sheets.
</p>
</body>
</html>

```

Margins and Padding: The margin properties are named margin, which applies to all four sides of an element: margin-left, margin-right, margin-top, and margin-bottom. The padding properties are named padding, which applies to all four sides: padding-left, padding-right, padding-top, and padding-bottom.

```

<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- marpads.html
An example to illustrate margins and padding
-->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head> <title> Margins and Padding </title>
<style type = "text/css">
p.one {margin: 0.2in;
padding: 0.2in;
background-color: #C0C0C0;
border-style: solid;
}
p.two {margin: 0.1in;
padding: 0.3in;
background-color: #C0C0C0;
border-style: solid;
}
p.three {margin: 0.3in;
padding: 0.1in;
background-color: #C0C0C0;
border-style: solid;
}

```



```

    p.four {margin:0.4in;
           background-color: #C0C0C0;}
    p.five {padding: 0.4in;
           background-color: #C0C0C0;
           }
</style>
</head>
<body>
  <p>
    Here is the first line.
  </p>
  <p class = "one">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.2in,
    padding = 0.2in]
  </p>
  <p class = "two">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.1in,
    padding = 0.3in]
  </p>
  <p class = "three">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.3in,
    padding = 0.1in]
  </p>
  <p class = "four">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [margin = 0.4in,
    no padding, no border]
  </p>
  <p class = "five">
    Now is the time for all good Web programmers to
    learn to use style sheets. <br /> [padding = 0.4in,
    no margin, no border]
  </p>
  <p>
    Here is the last line.
  </p>
</body>
</html>

```

10) Explain different ways of including CSS style information to a HTML doc.

LEVELS OF STYLE SHEETS

- The three levels of style sheets, in order from lowest level to highest level, are inline, document level, and external.
- Inline style sheets apply to the content of a single XHTML element.
- Document-level style sheets apply to the whole body of a document.
- External style sheets can apply to the bodies of any number of documents.
- Inline style sheets have precedence over document style sheets, which have precedence over external style sheets.
- Inline style specifications appear within the opening tag and apply only to the content of that tag.
- Document-level style specifications appear in the document head section and apply to the entire body of the document.
- External style sheets stored separately and are referenced in all documents that use them.
- External style sheets are written as text files with the MIME type text/css. They can be stored on any computer on the Web. The browser fetches external style sheets just as it fetches documents.
- The <link>tag is used to specify external style sheets. Within <link>, the rel attribute is used to specify the relationship of the linked-to document to the document in which the link appears. The href attribute of <link>is used to specify the URL of the style sheet document.

```
<link rel="stylesheet" type="text/css" href=http://www.cs.usc.edu/styles/wbook.css />
```

- The @import directive is an alternative way to use style specifications from other files.

@import url(filename);

Difference between link and @import:

- 1) @import can appear only at the beginning of the content of a style element
- 2) The imported files contains other markup, as well as style rules. Some import files can contain other import directives along with style rules.

STYLE SPECIFICATION FORMATS

The format of style specification depends on the level of stylesheet.

Inline Style Specification: appears as values of the style attribute of a tag, the general form is as follows:

Style = "Property1 : Value1; Property2 : Value2; Property3 : Value3; Property_n:Value_n;"

It is recommended that last property/value pair be followed by a semicolon.

Eg:

```
<h1 style ="font-family: 'Lucida Handwriting'; font-size: 50pt; color: Red;">Web Technology</h1>
```

Document Style Specification: appears as the content of a style element within the header of a document, general form of the content of a style element is as follows:

```
<style type = "text/css">
```

Rule list

</style>

Each style rule in a rule list has two parts: a selector, which indicates the tag or tags affected by the rule, and a list of property–value pairs. The list has the same form as the quoted list for inline style sheets, except that it is delimited by braces rather than double quotes. So, the form of a style rule is as follows:

Selector { Property1 : Value1; Property2 : Value2; Property3 : Value3;
Property_n:Value_n; }

Eg:

```
<style type = "text/css">
```

```
h1 {  
font-family: 'Lucida Handwriting';  
font-size: 50pt;  
color: Red;  
}
```

```
</style>
```

External Style Sheet: have a form similar to that of document style sheets. The external file consists of a list of style rules.

Eg

```
<head>
```

```
<link rel="stylesheet" type="text/css" href="cssfile.css">
```

```
</head>
```

```
Cssfile.css
```

```
P{  
Background-color:blue;  
Color:red;  
}
```